

Swift2.2中的新变化浅析

2016-03-28

0 个评论

来源: chaoyang805 的博客

收藏

我要投稿

Swift2.2已经更新了，这次更新去除了一些难用的语法还添加了一些缺失的特性，并且还弃用了一些有争议的语言特性。这篇文章将详细介绍Swift2.2中的一些主要的变化和和一些细微的改变，还会展示一些实际的代码例子来让你更快的上手Swift2.2。

1. ++ 和 -- 被弃用了

Swift 2.2正式地弃用了++和--操作符，意味着他们仍然可用但当你用到时会得到哦一条警告。弃用一般是完全移除的第一步，因此在这种情况下在Swift 3.0中这两个操作符将会被移除掉。

在使用这两个操作符的地方，你需要用+= 1和-= 1来替换掉他们。这两个操作符(+= -=)会一直存在，不会移除。

你可能会好奇为什么两个长期使用的操作符会被移除掉，尤其是他们存在于C、C#，Java以及调侃它的笑话 C++ 中时。下面列出了一些原因，但不是所有：

写++而非+= 1没有显著的节省时间虽然一旦你知道了它后用起来很容易，但是++对学习Swift的人来说并没有一个明确的含义，然而+=至少可以理解为”求和并赋值” C语言风格的for循环，这是++和-用的最多的地方，同样也被Swift弃用了，这也引出了我的下一点...

2. 传统的C风格for循环也被弃用了

对，你没看错：下面这种for循环将很快从Swift中完全移除掉：

```
for var i = 1; i <= 10; i += 1 {
    print("\(i) green bottles")
}
```

之所以称之为C风格的for循环，因为他们作为类C语言的特性已经存在了很长时间，概念上甚至在C语言之前都已经很久了。

虽然Swift也是(大概吧)类C语言，但他有一些比传统的for循环更新的、更聪明的替代品。结果就是：这个结构在Swift 2.2 中被弃用了并且”会在将来版本的Swift中移除掉”。

注意:当前的弃用警告并没有提示会在Swift 3.0移除，但我怀疑他会的。

要替换这些旧的for循环，使用其中一个替代选择即可。例如，上面的green bottles代码可以使用range的循环来重写，就像这样：

```
for i in 1...10 {
    print("\(i) green bottles")
}
```

记住，创建一个开始比结束值大的range是不对的，虽然你的代码会通过编译，但会在运行时崩溃掉。所以你要这么写：

儿童编程

- 文章 读书
- Win2000下关闭无用端口
 - 禁止非法用户登录综合设置 [win9x篇]
 - 关上可恶的后门——消除NetBIOS隐患
 - 网络入侵检测系统
 - 潜伏在Windows默认设置中的陷阱
 - 调制解调器的不安全
 - 构建Windows 2000服务器的安全防护林
 - SQL Server 2000的安全配置

重金招募VIP讲师

动画教程制作者(各类电脑技术)

邮箱订阅 红黑联盟 精彩内容

立即订阅

- 点击排行
- 使用python中的matplotlib进行绘图分析
 - 解决Android Studio加载第三方jar包，
 - HTML5实战与剖析之触摸事件(touchstar
 - spring security的原理及教程
 - Java 序列化Serializable详解（附详细
 - Android:Activity+Fragment及它们之间
 - iOS版本更新的App提交审核发布流程
 - Android ImageView的scaleType属性与

今日头条

儿童编程

软件工程师待遇

小黄人团队

python

前端工程师待遇

```
for i in 10...1 {
    print("\(i) green bottles")
}
```

而是应该写成这样：

```
for i in (1...10).reverse() {
    print("\(i) green bottles")
}
```

另一个选择是直接使用数组的快速枚举，就像这样：

```
var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
for number in array {
    print("\(number) green bottles")
}
```

3. 数组和其他的slice类型现在有removeFirst()函数了

removeFirst()方法对数组的操作一直很有用，但是直到现在一直缺少一个移除数组开始位置元素的对应方法。

好啦，添加了removeFirst()方法的Swift 2.2来拯救你了。这个方法会删除数组中的第一个元素并把它返回给你。

回头去看green bottles这个例子，你会发现两个坏习惯：第一，我使用了var而不是let，第二是打印了英语语法上的信息”一个绿瓶子们”。

这些都不是误写的，因为我要用它来演示removeFirst()了

```
var array = Array(1...10)
array.removeFirst()

for number in array {
    print("\(number) green bottles")
}
```

注意：虽然removeLast()有一个返回可选的等效方法popLast()，removeFirst()并没有类似的可选等效方法。这意味着如果你对一个空数组调用这个方法，你的代码会crash掉

4. 现在你可以比较元组了(在合适的格式下)

元组是一系列由逗号分隔的值，元组的值可以包含参数名也可以不包含，比如下面这个：

```
let singer = ("Taylor", "Swift")
let alien = ("Justin" "Bieber")
```

在旧版本的swift中，你不能比较两个元组的值，除非写一些笨重的代码，就像下面这样：

```
func == (t1: (T, T), t2: (T, T)) -> Bool {
    return t1.0 == t2.0 && t1.1 == t2.1
}
```

需要些这样的样板代码的要求不是非常友好的，当然这也只对只有两个值的元组有效。在Swift 2.2中，你不需要再写这样的代码了，因为元组可以直接进行比较了：

```
let singer = ("Taylor", "Swift")
```

- 1 Win10小马 原版镜像激活工具 永
- 2 KMS通用激活工具v2016.05.
- 3 Microsoft Toolkit (
- 4 C++ QT库开发
- 5 微信JS SDK开发，调用微信扫一扫
- 6 office2010激活工具
- 7 基于Android的计步器(Pedo
- 8 Java基础练习选择题(5)
- 9 win7永久激活码免费分享
- 10 黑苹果安装

```
let alien = ("Justin", "Bieber")

if singer == alien {
    print("Matching tuples!")
} else {
    print("Non-matching tuples!")
}
```

Swift 2.2的自动对比含两个元素的元组的效果就和我们刚写的函数功能一样。但它对六元元组同样适用，也就是有六个元素的元组。

至于为什么Swift的元组比较只支持到6元元组(而不是6百万)，这里有两个原因。首先，每一次额外的比较都会需要调用Swift标准库中的更多代码。另外，使用这么大的元组就是代码有问题了。应该换成使用结构体。

你可以通过改变上面的两个元组来看看元组是怎么进行比较的：

```
let singer = ("Taylor", 26)
let alien = ("Justin", "Bieber")
```

准备好接受Xcode长长的错误信息吧，但有趣的东西还在末尾呢：

```
note: overloads for '==' exist with these partially matching parameter lists: .....
((A, B), (A, B)), ((A, B, C), (A, B, C)), ((A, B, C, D), (A, B, C, D)), ((A, B, C, D, E)
```

5. 元组的splat语法被弃用了

让我们在元组上多聊一会：另一个被弃用的特性是Swift自从2010年(对，Swift发布以前)就存在的，它被称作“the tuple splat”，并没有多少人用过它。这也是被弃用的原因之一，虽然主要还是因为这个语法会给阅读代码带来歧义，所以才被弃用。

鉴于你可能会好奇，那我们来看一下，下面的splat语法的例子：

```
func describePerson(name: String, age: Int) {
    print("\(name) is \(age) years old")
}

let person = ("Taylor Swift", age: 26)

describePerson(persion)
```

要记住：别太喜欢你刚学的新知识，因为他们在Swift 2.2中被弃用了，并且会在未来的版本中完全移除掉

6. 更多的关键字可以用来当做参数标签了

参数标签是Swift中一个核心的特性，我们来撸一下下面这段代码：

```
for i in 1.stride(through: 9, by: 2) {
    print(i)
}
```

如果没了through和by标签，这段代码就失去了其自我注释的特性：

1.stride(9, 2)中的9和2是个啥意思？在这个例子中，Swift还使用了参数标签把1.stride(through: 9, by: 2)和1.stride(to: 9, by: 2)区分开来，这两段代码会产生不同的结果。

在Swift 2.2中，你可以用更多的语言关键字来作为参数标签了。你可能觉得为什么这会是件好事呢，考虑一下下面的代码：

```
func printGreeting(name: String, repeat repeatCount: Int) {  
    for _ in 0..  
repeatCount {  
        print(name)  
    }  
}  
  
printGreeting("Taylor", repeat: 5)
```

这里用到了repeat关键字作为参数标签，这个函数会重复打印一个字符串多次。因为repeat是一个关键字，这段代码在Swift 2.2之前是会出错的，你得用别的代替repeat，这让人很不爽。

注意：仍然有一些关键字是不能用作参数标签的，特殊的如var, let以及inout。

7. var 参数被弃用

另一个被抛弃的，同样也是有原因的：var参数被弃用是因为它提供的用处微不足道，并且经常和inout搞混。这东西太不易理解了，以至于我不得不加入到我的Swift language tests，但当你读到这里的时候我可能已经删除掉了。

为了掩饰，下面是用var关键字修改后的printGreeting()函数：

```
func printGreeting(var name: String, repeat repeatCount: Int) {  
    name = name.uppercaseString  
  
    for _ in 0..  
repeatCount {  
        print(name)  
    }  
}  
  
printGreeting("Taylor", repeat: 5)
```

与第一处不同的是前面的name变成了var name，并且name被换成了大写，所以会打印出” TAYLOR” 五次。

没有了var关键字，name就变成了常量，uppercaseString这一行就会出错。

var和inout之间的区别非常细微：使用var关键字你可以在函数体内部修改一个参数的值，而inout会让你所做的改变在函数结束后依然有效。

在Swift 2.2中，var被弃用了，并将在Swift 3.0 中移除。如果你正在使用这个，直接在函数内给这个变量做一份拷贝就可以了，比如：

```
func printGreeting(name: String, repeat repeatCount: Int) {  
    let upperName = name.uppercaseString  
  
    for _ in 0..  
repeatCount {  
        print(upperName)  
    }  
}  
  
printGreeting("Taylor", repeat: 5)
```

8.重命名了调试标识符：#line, #function, #file

Swift 2.1和更早的版本使用” screaming snake case” 的符号：FILE,LINE,COLUMN和FUNCTION，这些符号出现的地方会被编译器自动替换为文件名，行号，列号和函数名称

在Swift 2.2中，这些旧的符号已经被#file,#line,#column和#function替换掉了。你会相当熟悉如果已经使用了Swift 2.0的#available来检查iOS特性。正如Swift review官方所说的，它也引入了一条

出现#意味着会触发编译器的替代逻辑的规则。

下面我修改了printGreeting()函数，来看一下新旧两种调试标识符的用法：

```
func printGreeting(name: String, repeat repeatCount: Int) {  
    // old - deprecated!  
    print("This is on line \(__FILE__) of \(__FUNCTION__)" )  
  
    // new - shiny!  
    print("This is on line \(#line) of \(#function)")  
  
    let upperName = name.uppercaseString  
  
    for _ in 0..  
repeatCount {  
        print(upperName)  
    }  
}  
  
printGreeting("Taylor", repeat: 5)
```

多亏了代码补全，我应该加上你也可以使用#dsohandle了，但是如果你知道什么是dynamic shared object handles的话，你大概已经发现了这点变化了。

9. 字符串化的selector被弃用

Swift 2.2以前一个不受欢迎的功能是selector可以被写作字符串，像这样：

```
navigationItem.rightBarButtonItem = UIBarButtonItem(title: "Tap!", target: self, action:
```

如果你仔细看会发现，我写的是” buttonTaped” 而不是” buttonTapped”，但Xcode不会提示我出的错误，如果这个方法不存在的话。

这个问题在Swift 2.2中解决了：使用字符串作为selector被弃用了，并且你可以在上面的代码中这么写#selector(buttonTapped)，如果buttonTapped方法不存在的话，你会得到一条编译错误，这又是一个需要在类编译前消除的错误。

10. 编译时Swift 版本检查

Swift 2.2 加入了一个新的编译配置项使得用不同版本Swift写的代码联合编译成一个单独文件变得简单。这看起来可能是不必要的，但是考虑到使用Swift编写库的人们，他们能把目标版本设在Swift 2.2并且期待别人都用它或者目标版本设在Swift 2.0并期待用户能用Xcode升级吗？

使用新的编译选项能让你写两种不同口味的Swift，正确的那个会依据Swift编译器的版本来进行编译。

例如：

```
#if swift(>=2.2)  
print("Running Swift 2.2 or later")  
#else  
print("Running Swift 2.1 or earlier")  
#endif
```

就像#if os()编译选项一样，这个可以调整由编译器生成的代码：如果你正在使用Swift 2.2的编译器，第二个print()就不会被注意到。这意味着如果你愿意你可以随便胡扯点什么：

```
#if swift(>=2.2)  
print("Running Swift 2.2 or later")  
#else
```

```
THIS WILL COMPILE JUST FINE IF YOU ARE
USING A SWIFT 2.2 COMPILER BECAUSE
THIS BIT IS COMPLETELY IGNORED!

#endif
```

11. 新的文档关键字：recommended, recommended over, 和keyword

Swift支持MarkDown格式的注释来给你的代码添加额外的信息，所以你可以这么写：

```
/**
 Say hello to a specific person
 - parameters:
 - name: The name of the person to greet
 - returns: Absolutely nothing
 - authors:
 Paul Hudson
 Bilbo Baggins
 - bug: This is a deeply dull funtion
 */

func sayHello(name: String) {
    print("Hello, \(name)")
}

sayHello("Bob")
```

这些信息在代码补全中会使用到(“Say hello to a specific person” 会按照你书写的方式显示)并且在快速帮助中同样有用，其他的信息这时也会显示。

在Swift 2.2中，添加了三个新的关键字recommended,recommendedover和keyword.这几个关键字通过让你来指定哪个在Xcode中匹配的属性和方法应该被返回，从而使得代码补全更有用，但是现在看起来还没作用，所以这也只是一种猜测。

当它们突然可以用的时候，我希望你是像这么使用的：

```
/**
 - keyword: greeting
 - recommendedover: sayHelloToPaul
 */

func sayHello(name: String) { }

/**
 Always greets the same person
 - recommended: sayHello
 */


func sayHelloToPaul() { }
```

正如你看到的，recommended意思是更推荐另一个方法而不是这个，而recommendedover意思是更推荐这个方法而不是别的。

正如我所说，这在当前的Xcode 7.3版本中还是不可用的，但是我提交了一个bug给Apple，期待与能得到一些明确的说明这些关键字的作用，当我知道更多的时候我也会更新这个页面。

另一方面，Xcode 7.3有了新的代码补全特性：现在你可以打出几个字像是”strapp”来得到stringByAppendingString的高亮提示。或者”uitavc”来获UITableViewCell高亮提示，让你的脑袋适应这种文本快捷键还需要一些重新调整，但这确实给你编写代码的速度带来了显著的提升。





app开发报价单



php就业前景



深圳ui设计培训



平板电脑排行榜



支架游泳

[点击复制链接 与好友分享!](#)

[回本站首页](#)

上一篇: [Item 26: 避免对universal引用进行重载分析](#)

下一篇: [july算法课笔记分享](#)

相关文章

[Swift2编程之道：POP+MVVM架构](#)



消息推送领导品牌全面升级



极光推送

图文推荐



前端工程师待遇



支架泳池



swift编程语言



前端学习路线



百思不得姐之自定义C



关于聊天面板那点事~



runtime之消息 (三)



策略模式 (Strategy)

我有话说(0条评论)



来说两句吧...

微博登录

QQ登录

手机登录

[还没有评论, 快来抢沙发吧!](#)

红黑联盟正在使用畅言

yiya ui设计培训班 太阳能支架 竖屏 深圳ui设计培训 光伏支架 app外包 ui设计培训 林

工作就业 web前端 line是什么意思 ios设备 c语言培训班 美工学习 什么是前端开发 c++教程



安全工程师 软件工程师 网站工程师 网络工程师 电脑工程师

为新手量身定做的课程, 让菜鸟快速变身高手 正规公司助您腾飞



千豪直播平台的选择

七牛直播云

关闭